UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/798,916	03/11/2004	Scott J. Broussard	AUS920030818US1	7011

45502 7590 09/01/2009

DILLON & YUDELL LLP
8911 N. CAPITAL OF TEXAS HWY.,
SUITE 2110
AUSTIN, TX 78759

EXAMINER	
BROPHY, MATTHEW J	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
09/01/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/798,916	BROUSSARD, SCOTT J.	
	Examiner	Art Unit	
	MATTHEW J. BROPHY	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *5/27/2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4,8-14,18-24 and 28-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4,8-14,18-24 and 28-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to amendment field May 27, 2009.

2. Claims 1-4, 8-14, 18-24, 28-30 are pending.

Response to Amendment

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-4, 8, 9, 11-14, 18, 19, 21-24, 28 and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over as being anticipated by US PG Publication 2004/0078540 Cirne et al. hereinafter Cirne. In view of US Patent 7,089,460 Fu hereinafter Fu and further in US Patent 6,658,652 Alexander et al hereinafter Alexander.

Regarding Claims 1, 11 and 21 Cirne teaches: A method [system, or article of manufacture] for detecting memory leaks in a software program, said method comprising the steps of: monitoring a specified one or more analysis properties of software objects executing in the software program (**Cirne Paragraph "[0015] The present invention, roughly described, pertains to technology for identifying potential sources of memory leaks by tracking growth patterns of groups of stored items. One example of a group of stored items is an instance of a Java**

collection. If the growth pattern of a collection indicates that it may be the source of a memory leak, that collection is reported to a user and will continue to be tracked."), and identifying any software objects determined to have one or more analysis properties that exceeds that property's predetermined limit. **(Paragraph [0060] In step 322, it is determined whether the change counter is greater than the sensitivity counter. The sensitivity counter is a static number that corresponds to the sensitivity setting described above. For example, Table 2 shows that if the sensitivity setting is 10 then the sensitivity counter is 3, and if the sensitivity setting is 4 then the sensitivity counter is 7. Thus, the first time the first threshold is exceeded (e.g. where the threshold becomes 5.4); the change counter will equal 1, which is less than the sensitivity counter (step 332). Therefore, the collection is reported as not leaking in step 334. If the change counter is greater than the sensitivity counter, then the collection is reported as being a potential source of a leak in step 336. For example, if the sensitivity setting is 9, then the sensitivity setting will be 3. When the change counter is greater than 3, the collection will be reported as a potential source of a leak. In other words, when the size of the collection grows so that more than three thresholds have been exceeded, the collection is reported as being a potential source of a leak.").**

Cirne does not explicitly teach: determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property,

However, these limitations are taught by Fu: determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property **(Column 5, Lines 59-66, "FIG. 3 illustrates an exemplary cutoff weighting subroutine 300 that simply discards old memory usage elements. Memory usage weighting subroutine 300 begins at block 301 and proceeds to looping block 305 where an iteration through each usage data element begins. The first step in the loop is decision block 310 where a determination is made whether the current usage data element is older than a threshold time.")**. In addition it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Cirne with the object age comparison of Fu as: Cirne teaches the detection of memory leaks based on suspicious characteristics of object groups, wherein one of the characteristics is allocation time (see Cirne e.g. Paragraph [0052]). Also, Cirne teaches the comparison of another characteristic to a threshold (group size, referenced below). Finally, one of ordinary skill in the art would be motivated to combine the memory leak detection of Cirne with the object age threshold of Fu as the an object age above the threshold could be another indicator of a potential memory leak in the system of Cirne.

Cirne and Fu does not teach: generating a statistics report including the generated stack walkback for the at least one identified software object, wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that

indicates a location of an executing logic at a time of the out-of-memory error and wherein the generated statistics report identifies the location of at least one memory_ leak in the software program.

However, Alexander teaches:

generating a stack walkback for the at least one identified software object; **(Alexander Col. 17, Ln 38-53, "For example, at node 1152 in FIG. 11B, the call stack is CAB, and the statistics kept for this node are 2:3:4:1. Note that call stack CAB is first produced at time 2 in FIG. 10A, and is exited at time 3. Call stack CAB is produced again at time 4, and is exited at time 7. Thus, the first statistic indicates that this particular call stack, CAB, is produced twice in the trace. The second statistic indicates that call stack CAB exists for three units of time (at time 2, time 4, and time 6). The third statistic indicates the cumulative amount of time spent in call stack CAB and those call stacks invoked from call stack CAB (i.e., those call stacks having CAB as a prefix, in this case CABB). The cumulative time in the example shown in FIG. 11B is four units of time. Finally, the recursion depth of call stack CAB is one, as none of the three routines present in the call stack have been recursively entered.")** and

wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error **(Col. 3, Ln 9-13,"the memory may slowly lose portions of its allocable space until a "low" memory condition occurs in which no more memory**

may be allocated, which usually causes the system to crash soon after this condition arises. ") and wherein the generated statistics report identifies the location of at least one memory leak in the software program. **(Alexander III Col 18, Ln 12-34"Tracing may also be used to track memory allocation and deallocation. Every time a routine creates an object, a trace record could be generated. The tree structure of the present invention would then be used to efficiently store and retrieve information regarding memory allocation. Each node would represent the number of method calls, the amount of memory allocated within a method, the amount of memory allocated by methods called by the method, and the number of methods above this instance (i.e., the measure of recursion). Those skilled in the art will appreciate that the tree structure of the present invention may be used to represent a variety of performance data in a manner which is very compact, and allows a wide variety of performance queries to be performed.")**
In addition it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Cirne with the metrics of Alexander as Alexander allows the developer to analyze the statistics and trace record to debug a memory leak because "from such a report an analyst may notice an unexpected number of live objects of a particular class." (Alexander Col. 39, Ln 45-47.)

Regarding Claims 2, 12 and 22,: The rejections of claims 1, 11 and 21 are incorporated. Fu further teaches: wherein the one or more specified analysis properties include an object's age having a predetermined limit that is an object age limit; **(Column**

5, Lines 59-66, "FIG. 3 illustrates an exemplary cutoff weighting subroutine 300 that simply discards old memory usage elements. Memory usage weighting subroutine 300 begins at block 301 and proceeds to looping block 305 where an iteration through each usage data element begins. The first step in the loop is decision block 310 where a determination is made whether the current usage data element is older than a threshold time.")

and Cirne further teaches: and further comprising the step of calculating an object's age by timing a current period starting when the respective object was instantiated **(Paragraph [0053] "The entry in the log file created in step 266 includes the following information: current timestamp when written to the log, an identification (ID) for the collection, the class of the collection, the allocation time of the collection, allocation stack trace for the collection, current size of the collection and ten sample elements in the collection (represented by class name, followed by the toString( ) representation capped at 20 characters.")**.

Regarding Claims 3, 13 and 23, Cirne teaches: The limitations of claims 1, 11 and 21, further comprising the step of calculating object instance count growth as the magnitude of growth of an object's instance count over a given time period **(Paragraph [0051] "At the time for repeating the process, Agent 8 first checks the size of each collection (step 262). Each collection stores its own size. Agent 8 will sweep through all of its weak references and make sure that each object is still present in the heap by performing a simple null check. For each object that is still there,**

Agent 8 will read the size of that collection. In step 264, Agent 8 will update the heuristics for each collection for which it received a size. The heuristics (to be described in more detail below) determines whether the collection is a potential source of a leak or not.").

Regarding Claims 4, 14 and 24, Cirne teaches: The limitations of claims 1, 11 and 21, wherein the step of monitoring comprises monitoring objects within a class designated for monitoring **(Paragraph [0018] "One implementation of the present invention includes a method of monitoring for potential stores of memory leaks. The method includes tracking the size of a first group of stored items and determining whether that first group of stored items is a potential memory leak source based on change in size of the first group of stored items.")**.

Regarding Claims 8, 18 and 28, Cirne teaches: The limitations of claims 6, 16 and 26, further comprising the step of generating a web interface for user viewing of the statistics report at a computer display **(Paragraph [0032] "The workstations (e.g. 124 and 126) are the graphical user interface for viewing performance data.")**.

Regarding Claims 9, 19 and 29, Cirne teaches: The limitations of claims 1, 11 and 21, wherein the software objects are Java objects **(Cirne Paragraph "[0015] The present invention, roughly described, pertains to technology for identifying potential sources of memory leaks by tracking growth patterns of groups of stored items. One example of a group of stored items is an instance of a Java collection. If the growth pattern of a collection indicates that it may be the source**

of a memory leak, that collection is reported to a user and will continue to be tracked.").

3. Claims 10, 20 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over as being anticipated by US PG Publication 2004/0078540 Cirne et al. hereinafter Cirne. In view of US Patent 7,089,460 Fu hereinafter Fu and further in US Patent 6,658,652 Alexander et al hereinafter Alexander and further in view of US Patent 6,189,141 Benitez.

Regarding Claims 10, 20 and 30, Cirne teaches:

In addition Cirne further teaches: monitoring an amount of available memory for a software program referencing software objects **(Paragraph [0035] "A metric is a measurement of a specific application activity. Probes can be used to enable the reporting of a set of metrics for a managed application. Examples of metrics collected can include CORBA method timers, remote method indication method timers, thread counters, network bandwidth, JDBC update inquiry timers, servlet timers, Java Server Pages (JSP) timers, system logs, file system input and output bandwidth meters, availability and used memory, enterprise Java bean times, etc.")**; and upon such determination, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for a software

program referencing software objects dropping below an amount of available memory necessary to store a current stack walkback **(Paragraph [0049] "If leak detection is enabled and the time out period has not expired (step 206), then the code in the constructor for the collection object will create a stack trace for the collection object in step 208. In step 210, the code in the constructor for the collection object will pass a reference to the collection object and the stack trace to Agent 8." While the determination is not taught by Cirne as explained below, the storing of the current stack trace is inherently prior to reaching the threshold as Cirne allocates memory for the trace).**

Cirne does not explicitly teach: determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing software objects; upon determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software storing a current stack walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store [[a]] the current stack walkback.

However, Benitez teaches: determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing

software objects **(Benitez Col 38 Ln 66-Col 39, Ln5 "It is now assumed for illustrative purposes that storage locator 1210 has set the overflow flag. If the amount of memory made available by the elimination of cold traces, as described above, has been sufficient to reduce the amount of memory used in hot trace storage area 203 below the overflow threshold, cold trace detector and remover 1220 need not further identify cold traces for removal.")** upon-determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software storing a current stack walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store [[a]] the current stack walkback. **(Benitez Col 38 Ln 66-Col 39, Ln5 "It is now assumed for illustrative purposes that storage locator 1210 has set the overflow flag. If the amount of memory made available by the elimination of cold traces, as described above, has been sufficient to reduce the amount of memory used in hot trace storage area 203 below the overflow threshold, cold trace detector and remover 1220 need not further identify cold traces for removal.").**

In addition it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Benitez as Benitez use of thresholds allow the system to avoid memory overflows.

Response to Arguments

4. Applicant's arguments filed May 27, 2009 have been fully considered but they are not persuasive.

In remarks, Applicant Argues:

> The Office Action rejects independent claims 1, 11, and 21 under 103(a) as unpatentable over U.S. Patent App. Pub. No. 2004/007854 to Cirne et al. ("Cirne"), in view of U.S. Patent No. 7,089,460 to Fu ("Fu"), and further in view of U.S. Patent No. 6,658,652 to Alexander III et al. ("Alexander"). Amended Claims 1, 11, and 21 recite, in part:
> "...generating a statistics report including the generated stack walkback for the at least one identified software object, wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error, and wherein the generated statistics report identifies the location of at least one memory leak in the software program." However, neither Cirne, Fu, Alexander nor their combination teach or suggest the above-quoted claim limitations. First, the Examiner concedes on page 6 of the Office Action that:
> "Cime and Fu does not teach: generating a statistics report including the generated stack walkback for the at least one identified software object wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error and wherein the generated statistics report identifies the likely location of at least one memory_ [sic] leak in the software program."
> Thus, the Examiner relies solely on the Alexander reference in rejecting the above claim limitations recited in Claims 1, 11, and 21. In particular, the Examiner has cited two separate sections within the Alexander reference: (a) col. 17, lines 38-53; FIG. 11B (hereinafter "section (a)"); and (b) col. 18, lines 12-34 (hereinafter "section (b)").
> The Examiner asserts that the generation of a stack walkback is taught by col. 17, lines 38-53 and FIG. 11B of Alexander. However, Applicant respectfully disagrees. Section (a) of Alexander does not teach the generation of a statistics report that includes the generation of a stack walkback. As defined by ~[0029] of Applicant's Specification: "a stack walkback (also known as a Java stack trace) is a user-friendly snapshot of 'threads' and 'monitors' executing in a JVM. A stack walkback is used to track the history of an object instance to determine the point it was

created."
In contrast, Alexander teaches the generation of a call stack tree, which reflects call stacks observed during a system execution. Referring to FIG. 11B of Alexander, the call stack tree is comprised of nodes in which statistics are kept for each node in the call stack tree. Each node in the call stack tree represents a function entry point (Alexander, col. 16, 62-65). A stack walkback, as defined by Applicant's claimed invention, is not focused on function entry point(s), but rather on threads and monitors executing in a Java Virtual Machine (JVM).

Examiner's Response:

5. Examiner respectfully disagrees. In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., "a user-friendly snapshot of 'threads' and 'monitors' executing in a JVM") are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993). Here, the examiner interprets the "stack walkback" as a java stack trace equivalent to the call stack tree 1100 of Alexander which is generated from trace data. Additionally, the call stack tree is presented in a pictorial view as shown in FIG. 11B. Therefore, this rejection is maintained.

In Remarks, Applicant Argues:

The Examiner next asserts that section (b) of Alexander teaches that a statistics report is generated in a format that indicates a location of an executing logic at a time of an out-of- memory error and identifies the likely location of at least one memory leak in the software program. Again, Applicant respectfully disagrees. The tree structure taught by Alexander is used to track memory allocation and deallocation metrics associated with object processing initiated on behalf of an executing method. For example,

Alexander describes a call stack tree which reflects call stacks observed during a specific example of system execution. At each node in the call stack tree, several statistics are recorded:
1.) the number of distinct times the call stack is produced,
2.) the sum of the time spent in the call stack,
3.) the total time spent in the call stack plus the time in those call stacks invoked from this call stack (referred to as cumulative time), and
4.) the number of instances of a routine above a particular instance (indicating a recursion depth)
While Alexander tracks performance data, it should be noted that statistics 1.) through 4.) in Alexander are all specifically stated to be "time-based statistics" (Alexander, col. 17, lines 26- 37). Alexander fails to teach or suggest a location of an executing logic at a time of an out-of- memory error. At best, Alexander is used to merely detect the presence of a memory leak.
Since Cirne, Fu, and Alexander independently fail to teach or suggest the limitations recited in amended Claims 1, 11, and 21, the references cannot be properly combined for purposes of § 103(a) to teach or suggest Applicant's claimed invention.

Examiner's Response:

Examiner respectfully disagrees. First, the trace records saved in the call stack tree include among other things the time, the event and the name of the executing module (Step 1352, FIG. 13) which the examiner interprets as indicating "a location of an executing logic". Additionally, as quoted above, Alexander teaches that the execution terminates with a "system crash" when the system is out of memory. (Col. 3, Ln 9-13) Therefore, the tracing of the execution is completed prior to the system crash, and the format (see FIGs 11A & 11B) that presents the type of each trace event, up to the termination of execution.

In Remarks, Applicant Argues:

> Dependent Claims 10, 20~ and 30
> The Office Action rejects dependent claims 10, 20, and 30 under 103(a) as unpatentable over Cirne, in view of Fu, further in view of Alexander, and further still in view of U.S. Patent No. 6,189,141 to Benitez et al. ("Benitez").
> Claim 10 (and similarly claims 20 and 30) recite, in part: "upon determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback."
> The Examiner has cited Benitez to assert that the reference teaches the above cited limitation. However, Applicant respectfully disagrees. Benitez teaches that when an overflow condition is present (i.e., a hot trace storage area is becoming full), cold traces within the hot trace storage area are removed to make storage space for additional hot traces; see col. 35, lines 29-40 of Benitez). Benitez does not teach storing a current stack walkback when it is determined that the amount of available memory is within a threshold, as recited in claims 10, 20 and 30. Thus, in Benitez, the

cold traces that were previously stored in the hot storage area are not stored. Rather, the cold traces are removed altogether, according to Benitez.

Examiner's Response:

Examiner respectfully disagrees. As previously described in response to this same argument in the prior office action: Benetiz teaches "upon determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software storing a current stack walkback of currently referenced software objects prior to the amount of available memory for the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback" because: Benetiz teaches "determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available" by determining when an overflow condition exists and "storing a current stack walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback" by storing hot traces in the hot trace area until the overflow happens. (Benitez Col 38, Ln 66-Col 39, Ln 5).

Conclusion

6. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to MATTHEW J. BROPHY whose telephone number is 571-270-1642. The examiner can normally be reached on Monday-Thursday 8:00AM-5:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MJB

8/26/2009

/Ted T. Vo/
Primary Examiner, Art Unit 2191